

Отчет по лабораторной работе №9

Понятие подпрограммы. Отладчик GDB.

Паласиос Фелипе

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	8
4	Выводы	20
	Список литературы	21

Список иллюстраций

3.1	каталог	8
3.2	исполняемый файл	9
3.3	добавив подпрограмму _subcalcul	9
3.4	lab09-2	10
3.5	lab09-2.run	10
3.6	_start	11
3.7	disassemble __start	11
3.8	Название рисунка	12
3.9	layout	13
3.10	содержимое регистров	14
3.11	значение переменной msg1 по имени	14
3.12	Изменен первый символ переменной msg1	15
3.13	команда set	15
3.14	quit	16
3.15	lab09-3.asm.ключ -arg	17
3.16	установим точку	17
3.17	позиции стека	18
3.18	1	18

Список таблиц

1 Цель работы

Приобретение навыков написания программ с использованием подпрограмм. Знакомство методами отладки при помощи GDB и его основными возможностями.

2 Задание

1. Создайте каталог для выполнения лабораторной работы No 9, перейдите в него и создайте файл lab09-1.asm
 2. 1.Введите в файл lab09-1.asm текст программы из листинга 9.1. Создайте исполняемый файл и проверьте его работу.

2.Замените текст программы, добавив подпрограмму `_subcalcul` в подпрограмму `_calcul`, для вычисления выражения $\text{X}(\text{X}(\text{X}))$, где X вводится с клавиатуры, $\text{X}(\text{X}) = 2\text{X} + 7$, $\text{X}(\text{X}) = 3\text{X} - 1$.
 3. Создайте файл lab09-2.asm с текстом программы из Листинга 9.2. (Программа печати сообщения Hello world!). 1.Проверьте работу программы, запустив ее в оболочке GDB с помощью команды `run` (сокращённо `r`) 2.для более подробного анализа программы установите брейкпоинт на метку `_start`, с которой начинается выполнение любой ассемблерной программы, и запустите её.
 - 3.Посмотрите дисассимилированный код программы с помощью команды `disassemble` начиная с метки `_start`
 - 4.Переключитесь на отображение команд с Intel'овским синтаксисом, введя команду `set disassembly-flavor intel`
 - 5.Включите режим псевдографики для более удобного анализа программы
4. Работа с данными программы в GDB

1.Посмотреть содержимое регистров также можно с помощью команды `info registers` (или `i r`)

2.Посмотрите значение переменной `msg1` по имени (`gdb`) `x/1sb &msg1`
`0x804a000 : "Hello,"`

3.Посмотрите дисассимилированный код программы с помощью команды `disassemble` начиная с метки `_start`

4.С помощью команды `set` измените значение регистра `ebx`

5. Скопируйте файл `lab8-2.asm`, созданный при выполнении лабораторной работы No8, с программой выводящей на экран аргументы командной строки (Листинг 8.2) в файл с именем `lab09-3.asm`. Создайте исполняемый файл. Для загрузки в `gdb` программы с аргументами необходимо использовать ключ `-args`. Загрузите исполняемый файл в отладчик, указав аргументы

1.Исследуем расположение аргументов командной строки в стеке после запуска программы с помощью `gdb`. Для начала установим точку останова перед первой инструкцией в программе и запустим ее. (`gdb`) `b _start` (`gdb`) `run`

2.Адрес вершины стека храниться в регистре `esp` и по этому адресу располагается число равное количеству аргументов командной строки (включая имя программы) Посмотрите остальные позиции стека – по адресу `[esp+4]` располагается адрес в памяти где находится имя программы, по адресу `[esp+8]` храниться адрес первого аргумента, по адресу `[esp+12]` – второго и т.д

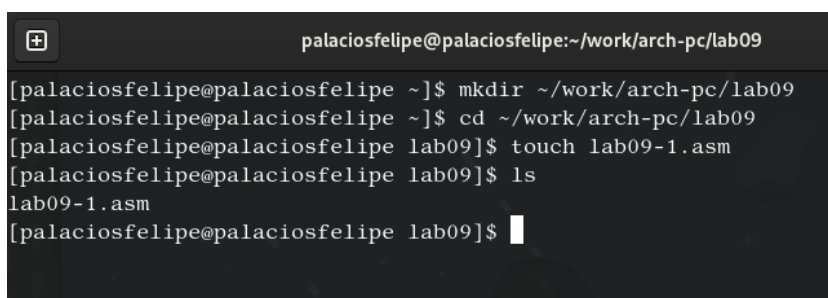
6. Задание для самостоятельной работы

1.Преобразуйте программу из лабораторной работы No8 (Задание No1 для самостоятельной работы), реализовав вычисление значения функции $\text{fib}(n)$ как подпрограмму

2.В листинге 9.3 приведена программа вычисления выражения $(3 + 2) \times 4 + 5$. При запуске данная программа дает неверный результат. Проверьте это. С помощью отладчика GDB, анализируя изменения значений регистров, определите ошибку и исправьте ее.

3 Выполнение лабораторной работы

1. Создайте каталог для выполнения лабораторной работы No 9, перейдите в него и создайте файл lab09-1.asm (рис. 3.1).

A terminal window with a dark background. The title bar shows a plus icon and the text 'palaciosfelipe@palaciosfelipe:~/work/arch-pc/lab09'. The terminal contains the following commands and output:

```
[palaciosfelipe@palaciosfelipe ~]$ mkdir ~/work/arch-pc/lab09
[palaciosfelipe@palaciosfelipe ~]$ cd ~/work/arch-pc/lab09
[palaciosfelipe@palaciosfelipe lab09]$ touch lab09-1.asm
[palaciosfelipe@palaciosfelipe lab09]$ ls
lab09-1.asm
[palaciosfelipe@palaciosfelipe lab09]$
```

Рис. 3.1: каталог

2. 1. Введите в файл lab09-1.asm текст программы из листинга 9.1. Создайте исполняемый файл и проверьте его работу (рис. 3.2).


```
palaciosfelipe@palaciosfelipe:~/work/arch-pc/lab09 — ./lab09-1
[palaciosfelipe@palaciosfelipe ~]$ mkdir ~/work/arch-pc/lab09
[palaciosfelipe@palaciosfelipe ~]$ cd ~/work/arch-pc/lab09
[palaciosfelipe@palaciosfelipe lab09]$ touch lab09-1.asm
[palaciosfelipe@palaciosfelipe lab09]$ ls
lab09-1.asm
[palaciosfelipe@palaciosfelipe lab09]$ nasm -f elf lab09-1.asm
lab09-1.asm:1: warning: unterminated string [-w+other]
lab09-1.asm:1: error: unable to open include file 'in_out.asm': No such file or
directory
[palaciosfelipe@palaciosfelipe lab09]$ nasm -f elf lab09-1.asm
lab09-1.asm:1: error: unable to open include file 'in_out.asm': No such file or
directory
[palaciosfelipe@palaciosfelipe lab09]$ nasm -f elf lab09-1.asm
lab09-1.asm:1: error: unable to open include file 'in_out.asm': No such file or
directory
[palaciosfelipe@palaciosfelipe lab09]$ nasm -f elf lab09-1.asm
[palaciosfelipe@palaciosfelipe lab09]$ ld -m elf_i386 -o lab09-1 lab09-1.o
[palaciosfelipe@palaciosfelipe lab09]$ ./lab09-1
Введите x: 
```

Рис. 3.2: исполняемый файл

2.Измените текст программы, добавив подпрограмму `_subcalcul` в подпрограмму `_calcul`

```
lab09-1.asm
    .calcul:
    push ebx
    mov ebx,2
    mul ebx
    add eax, 7
    pop ebx
    ret

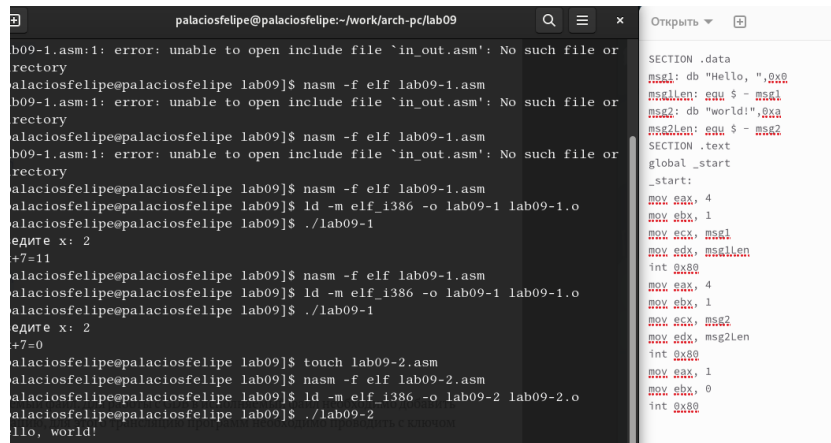
    _subcalcul:
    push ebx
    mov ebx, 3
    mul ebx
    dec ebx
    mov [esi],eax

    ret ; выход из подпрограммы

palaciosfelipe@palaciosfelipe ~]$ mkdir ~/work/arch-pc/lab09
palaciosfelipe@palaciosfelipe ~]$ cd ~/work/arch-pc/lab09
palaciosfelipe@palaciosfelipe lab09]$ touch lab09-1.asm
palaciosfelipe@palaciosfelipe lab09]$ ls
lab09-1.asm
palaciosfelipe@palaciosfelipe lab09]$ nasm -f elf lab09-1.asm
lab09-1.asm:1: warning: unterminated string [-w+other]
lab09-1.asm:1: error: unable to open include file 'in_out.asm': No such file or
directory
palaciosfelipe@palaciosfelipe lab09]$ nasm -f elf lab09-1.asm
lab09-1.asm:1: error: unable to open include file 'in_out.asm': No such file or
directory
palaciosfelipe@palaciosfelipe lab09]$ nasm -f elf lab09-1.asm
lab09-1.asm:1: error: unable to open include file 'in_out.asm': No such file or
directory
palaciosfelipe@palaciosfelipe lab09]$ nasm -f elf lab09-1.asm
palaciosfelipe@palaciosfelipe lab09]$ ld -m elf_i386 -o lab09-1 lab09-1.o
palaciosfelipe@palaciosfelipe lab09]$ ./lab09-1
Введите x: 2
x^2=11
palaciosfelipe@palaciosfelipe lab09]$ nasm -f elf lab09-1.asm
palaciosfelipe@palaciosfelipe lab09]$ ld -m elf_i386 -o lab09-1 lab09-1.o
palaciosfelipe@palaciosfelipe lab09]$ ./lab09-1
Введите x: 
```

Рис. 3.3: добавив подпрограмму `_subcalcul`

3. Создайте файл `lab09-2.asm` с текстом программы из Листинга 9.2. (Программа печати сообщения `Hello world!`) (рис. 3.4).

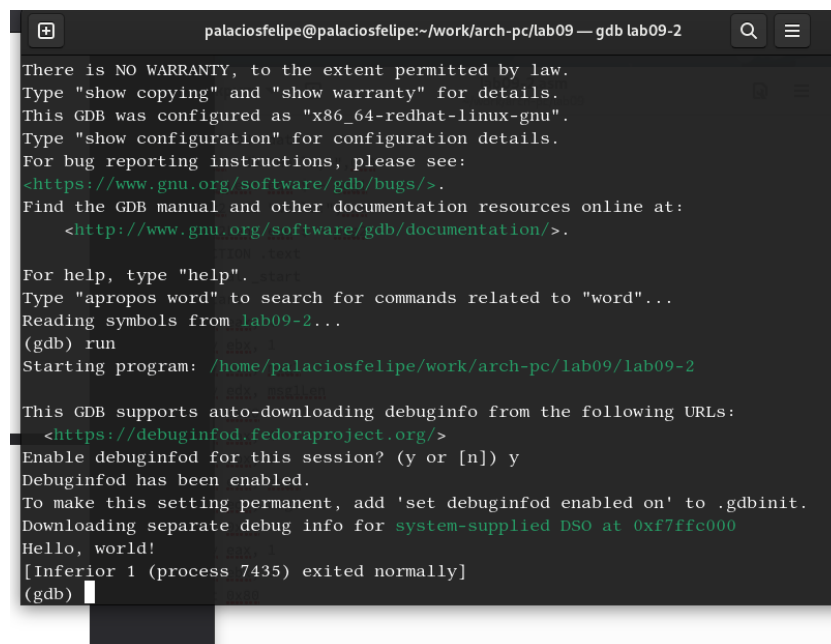


```
palaciosfelipe@palaciosfelipe:~/work/arch-pc/lab09
b09-1.asm:1: error: unable to open include file `in_out.asm': No such file or
rectory
alaciosfelipe@palaciosfelipe lab09]$ nasm -f elf lab09-1.asm
b09-1.asm:1: error: unable to open include file `in_out.asm': No such file or
rectory
alaciosfelipe@palaciosfelipe lab09]$ nasm -f elf lab09-1.asm
b09-1.asm:1: error: unable to open include file `in_out.asm': No such file or
rectory
alaciosfelipe@palaciosfelipe lab09]$ nasm -f elf lab09-1.asm
alaciosfelipe@palaciosfelipe lab09]$ ld -m elf_i386 -o lab09-1 lab09-1.o
alaciosfelipe@palaciosfelipe lab09]$ ./lab09-1
едите x: 2
+7=11
alaciosfelipe@palaciosfelipe lab09]$ nasm -f elf lab09-1.asm
alaciosfelipe@palaciosfelipe lab09]$ ld -m elf_i386 -o lab09-1 lab09-1.o
alaciosfelipe@palaciosfelipe lab09]$ ./lab09-1
едите x: 2
+7=0
alaciosfelipe@palaciosfelipe lab09]$ touch lab09-2.asm
alaciosfelipe@palaciosfelipe lab09]$ nasm -f elf lab09-2.asm
alaciosfelipe@palaciosfelipe lab09]$ ld -m elf_i386 -o lab09-2 lab09-2.o
alaciosfelipe@palaciosfelipe lab09]$ ./lab09-2
Hello, world!
```

```
SECTION .data
msg1: db "Hello, ",0x0
msg2: db "world!",0x0
SECTION .text
global _start
_start:
mov $x, 4
mov $x, 1
mov $x, msg1
mov $x, msg2len
int 0x80
mov $x, 4
mov $x, 1
mov $x, msg2
mov $x, msg2len
int 0x80
mov $x, 1
mov $x, 0
int 0x80
```

Рис. 3.4: lab09-2

1.Проверьте работу программы, запустив ее в оболочке GDB с помощью коман-
ды run (сокращённо r) (рис. 3.5).



```
palaciosfelipe@palaciosfelipe:~/work/arch-pc/lab09 — gdb lab09-2
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab09-2...
(gdb) run
Starting program: /home/palaciosfelipe/work/arch-pc/lab09/lab09-2

This GDB supports auto-downloading debuginfo from the following URLs:
<https://debuginfod.fedoraproject.org/>
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
Downloading separate debug info for system-supplied DSO at 0xf7ffc000
Hello, world!
[Inferior 1 (process 7435) exited normally]
(gdb)
```

Рис. 3.5: lab09-2. run

2.для более подробного анализа программы установите брейкпоинт на метку
_start, с которой начинается выполнение любой ассемблерной программы, и
запустите её (рис. 3.6).

```
palaciosfelipe@palaciosfelipe:~/work/arch-pc/lab09 — gdb lab09-2
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab09-2...
(gdb) run
Starting program: /home/palaciosfelipe/work/arch-pc/lab09/lab09-2
This GDB supports auto-downloading debuginfo from the following URL
<https://debuginfod.fedoraproject.org/>
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to
Downloading separate debug info for system-supplied DSO at 0xf7ff
Hello, world!
[Inferior 1 (process 7435) exited normally]
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab09-2.asm, line 9.
(gdb) run
Starting program: /home/palaciosfelipe/work/arch-pc/lab09/lab09-2
Breakpoint 1, _start () at lab09-2.asm:9
9      mov eax, 4
(gdb)
```

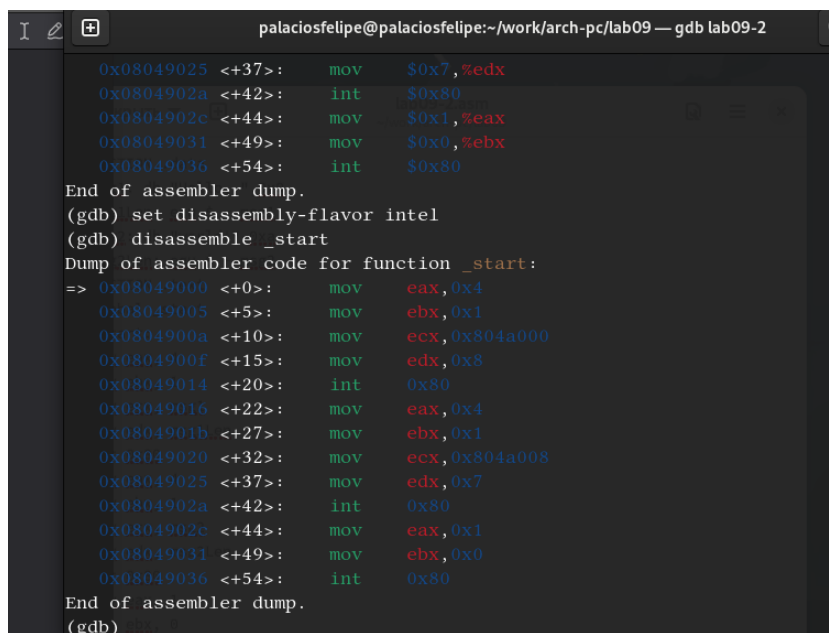
Рис. 3.6: `_start`

3.Посмотрите дисассимилированный код программы с помощью команды `disassemble` начиная с метки `_start` (рис. 3.7).

```
palaciosfelipe@palaciosfelipe:~/work/arch-pc/lab09 — gdb lab09-2
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab09-2.asm, line 9.
(gdb) run
Starting program: /home/palaciosfelipe/work/arch-pc/lab09/lab09-2
Breakpoint 1, _start () at lab09-2.asm:9
9      mov eax, 4
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x08049000 <+0>:    mov     $0x4,%eax
0x08049005 <+5>:    mov     $0x1,%ebx
0x0804900a <+10>:   mov     $0x804a000,%ecx
0x0804900f <+15>:   mov     $0x8,%edx
0x08049014 <+20>:   int     $0x80
0x08049016 <+22>:   mov     $0x4,%eax
0x0804901b <+27>:   mov     $0x1,%ebx
0x08049020 <+32>:   mov     $0x804a008,%ecx
0x08049025 <+37>:   mov     $0x7,%edx
0x0804902a <+42>:   int     $0x80
0x0804902c <+44>:   mov     $0x1,%eax
0x08049031 <+49>:   mov     $0x0,%ebx
0x08049036 <+54>:   int     $0x80
End of assembler dump.
(gdb)
```

Рис. 3.7: `disassemble _start`

4.Переключитесь на отображение команд с Intel'овским синтаксисом, введя команду `set disassembly-flavor intel` (рис. 3.8).



```
palaciosfeliipe@palaciosfeliipe:~/work/arch-pc/lab09 — gdb lab09-2
0x08049025 <+37>:  mov    $0x7,%edx
0x0804902a <+42>:  int     $0x80
0x0804902c <+44>:  mov    $0x1,%eax
0x08049031 <+49>:  mov    $0x0,%ebx
0x08049036 <+54>:  int     $0x80
End of assembler dump.
(gdb) set disassembly-flavor intel
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x08049000 <+0>:  mov    eax,0x4
      0x08049005 <+5>:  mov    ebx,0x1
      0x0804900a <+10>: mov    ecx,0x804a000
      0x0804900f <+15>: mov    edx,0x8
      0x08049014 <+20>: int     0x80
      0x08049016 <+22>: mov    eax,0x4
      0x0804901b <+27>: mov    ebx,0x1
      0x08049020 <+32>: mov    ecx,0x804a008
      0x08049025 <+37>: mov    edx,0x7
      0x0804902a <+42>: int     0x80
      0x0804902c <+44>: mov    eax,0x1
      0x08049031 <+49>: mov    ebx,0x0
      0x08049036 <+54>: int     0x80
End of assembler dump.
(gdb)
```

Рис. 3.8: Название рисунка

5.Включите режим псевдографики для более удобного анализа программы (рис. 3.9).

```
palaciosfelipe@palaciosfelipe:~/work/arch-pc/lab09 — gdb lab09-2

[ Register Values Unavailable ]

B+> 0x8049000 <_start> mov eax,0x4
0x8049005 <_start+5> mov ebx,0x1
0x804900a <_start+10> mov ecx,0x804a000
0x804900f <_start+15> mov edx,0x8
0x8049014 <_start+20> int 0x80
0x8049016 <_start+22> mov eax,0x4

native process 7675 In: _start L9 PC: 0x8049000
(gdb) layout regs
(gdb)
```

Рис. 3.9: layout

4. Работа с данными программы в GDB

1.Посмотреть содержимое регистров также можно с помощью команды `info registers` (или `i r`) (рис. 3.10).

```

palaciosfelipe@palaciosfelipe:~/work/arch-pc/lab09 — gdb lab09-2

[ Register Values Unavailable ]

B+> 0x8049000 <_start>      mov     eax,0x4
0x8049005 <_start+5>      mov     ebx,0x1
0x804900a <_start+10>     mov     ecx,0x804a000
0x804900f <_start+15>     mov     edx,0x8
0x8049014 <_start+20>     int     0x80
0x8049016 <_start+22>     mov     eax,0x4

native process 7675 In: _start                                L9    PC: 0x8049000
eax      0x0 0x0 0x0 0x0 0
ecx      0x0 0x0 0x1 0x0 0
edx      0x0 0x0 0x0 0x0 0
ebx      0x1 0x0 0x0 0x0 0
esp      0x0 0xffffd060 0xffffd060
ebp      0x0 0x0 0x0 0x0 0
esi      0x0 0x0 0x0 0x0 0
--Type <RET> for more, q to quit, c to continue without paging--

```

Рис. 3.10: содержимое регистров

2.Посмотрите значение переменной msg1 по имени (gdb) x/1sb &msg1
0x804a000 : “Hello,” (рис. 3.11).

```

palaciosfelipe@palaciosfelipe:~/work/arch-pc/lab09 — gdb lab09-2

[ Register Values Unavailable ]

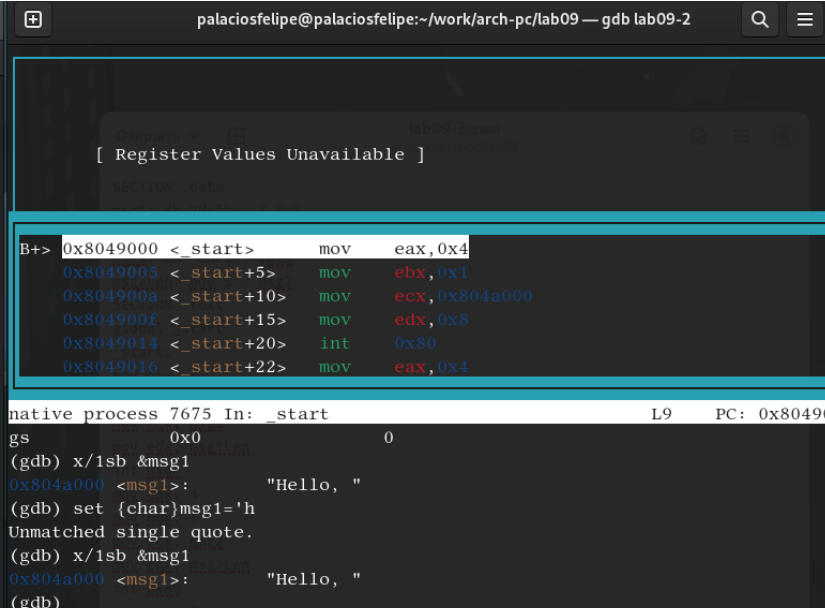
B+> 0x8049000 <_start>      mov     eax,0x4
0x8049005 <_start+5>      mov     ebx,0x1
0x804900a <_start+10>     mov     ecx,0x804a000
0x804900f <_start+15>     mov     edx,0x8
0x8049014 <_start+20>     int     0x80
0x8049016 <_start+22>     mov     eax,0x4

native process 7675 In: _start                                L9    PC: 0x80490
ss      0x2b 0x0 0x0 0x0 43
ds      0x2b 0x0 0x0 0x0 43
es      0x2b 0x0 0x0 0x0 43
fs      0x0 0x0 0x0 0x0 0
gs      0x0 0x0 0x0 0x0 0
(gdb) x/1sb &msg1
0x804a000 <msg1>:      "Hello, "
(gdb)

```

Рис. 3.11: значение переменной msg1 по имени

3.Измените первый символ переменной msg1 (рис. 3.12).



The screenshot shows a GDB session with the following assembly code loaded:

```
B+> 0x8049000 <_start> mov eax,0x4
0x8049005 <_start+5> mov ebx,0x1
0x804900a <_start+10> mov ecx,0x804a000
0x804900f <_start+15> mov edx,0x8
0x8049014 <_start+20> int 0x80
0x8049016 <_start+22> mov eax,0x4
```

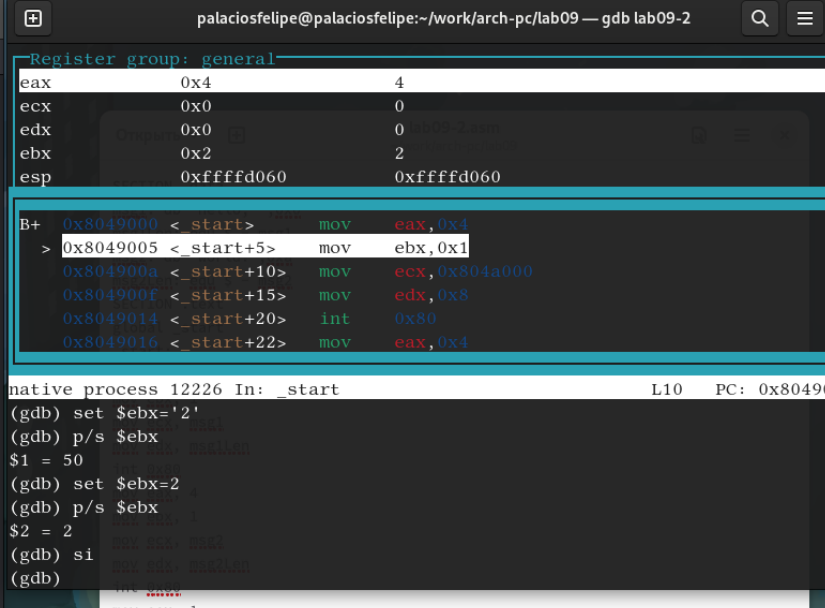
The memory dump for variable msg1 is shown below:

```
(gdb) x/1sb &msg1
0x804a000 <msg1>: "Hello, "
```

The user has entered the command `(gdb) set {char}msg1='h'`, which resulted in an "Unmatched single quote." error. The memory dump still shows "Hello, ".

Рис. 3.12: Изменен первый символ переменной msg1

4.С помощью команды set измените значение регистра ebx (рис. 3.13).



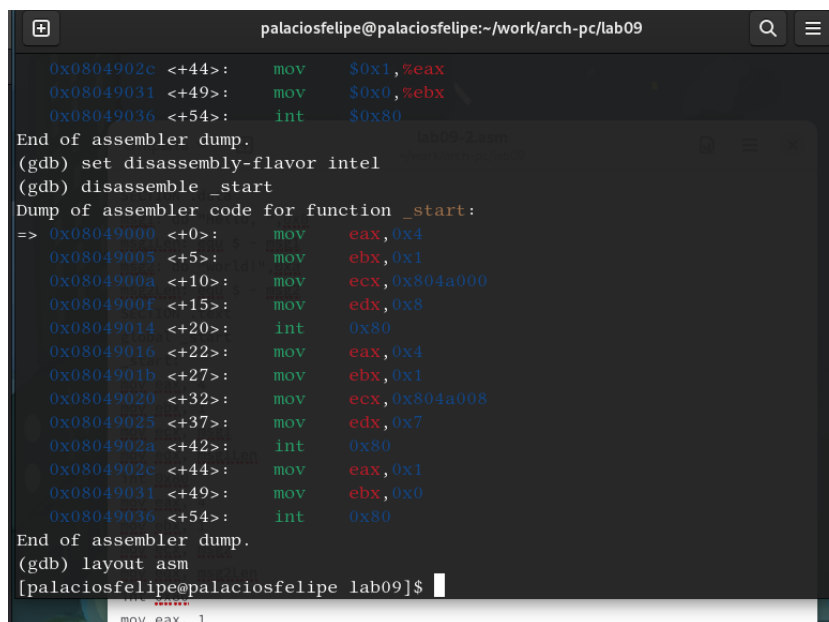
The screenshot shows a GDB session with the following register values:

Register	Value
eax	0x4
ecx	0x0
edx	0x0
ebx	0x2
esp	0xffffd060

The assembly code is the same as in Figure 3.12. The user has entered the command `(gdb) set $ebx='2'`, which resulted in an error. The user then entered `(gdb) set $ebx=2`, which was successful. The register value for ebx is now 2.

Рис. 3.13: команда set

5. Завершите выполнение программы с помощью команды `continue` (сокращенно `c`) или `stepi` (сокращенно `si`) и выйдите из GDB с помощью команды `quit` (сокращенно `q`). (рис. 3.14).



```
palaciosfelipe@palaciosfelipe:~/work/arch-pc/lab09
0x0804902c <+44>:  mov    $0x1,%eax
0x08049031 <+49>:  mov    $0x0,%ebx
0x08049036 <+54>:  int     $0x80
End of assembler dump.
(gdb) set disassembly-flavor intel
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x08049000 <+0>:  mov    eax,0x4
0x08049005 <+5>:    mov    ebx,0x1
0x0804900a <+10>:   mov    ecx,0x804a000
0x0804900f <+15>:   mov    edx,0x8
0x08049014 <+20>:   int     0x80
0x08049016 <+22>:   mov    eax,0x4
0x0804901b <+27>:   mov    ebx,0x1
0x08049020 <+32>:   mov    ecx,0x804a008
0x08049025 <+37>:   mov    edx,0x7
0x0804902a <+42>:   int     0x80
0x0804902c <+44>:   mov    eax,0x1
0x08049031 <+49>:   mov    ebx,0x0
0x08049036 <+54>:   int     0x80
End of assembler dump.
(gdb) layout asm
[palaciosfelipe@palaciosfelipe lab09]$
mov eax, 1
```

Рис. 3.14: quit

5. Скопируйте файл `lab8-2.asm`, созданный при выполнении лабораторной работы No8, с программой выводящей на экран аргументы командной строки (Листинг 8.2) в файл с именем `lab09-3.asm`. Создайте исполняемый файл. Для загрузки в `gdb` программы с аргументами необходимо использовать ключ `-args`. Загрузите исполняемый файл в отладчик, указав аргументы (рис. 3.15).


```
palaciosfelipe@palaciosfelipe:~/work/arch-pc/lab09 — gdb --args lab09-3 аргумент1 а...
Reading symbols from lab09-3...
(No debugging symbols found in lab09-3)
(gdb)
[1]+  Остановлен   gdb --args lab09-3 аргумент1 аргумент 2 'аргумент 3'
palaciosfelipe@palaciosfelipe lab09]$ gdb --args lab09-3 аргумент1 аргумент 2 '
аргумент 3'
GNU gdb (GDB) Fedora Linux 13.1-1.fc37
Copyright (C) 2023 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab09-3...
(No debugging symbols found in lab09-3)
(gdb)
```

Рис. 3.15: lab09-3.asm.ключ -arg

1.Исследуем расположение аргументов командной строки в стеке после запуска программы с помощью gdb.Для начала установим точку останова перед первой инструкцией в программе и запустим ее (gdb) b _start (gdb) run (рис. 3.16).

```
palaciosfelipe@palaciosfelipe:~/work/arch-pc/lab09 — gdb --args lab09-3 аргумент1 а...
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab09-3...
(No debugging symbols found in lab09-3)
(gdb) b _start
Breakpoint 1 at 0x80490e8
(gdb) run
Starting program: /home/palaciosfelipe/work/arch-pc/lab09/lab09-3 аргумент1 аргу
мент 2 аргумент\ 3
This GDB supports auto-downloading debuginfo from the following URLs:
<https://debuginfod.fedoraproject.org/>
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
Downloading separate debug info for system-supplied DSO at 0xf7ffc000
Breakpoint 1, 0x080490e8 in _start ()
(gdb)
```

Рис. 3.16: установим точку

2.Адрес вершины стека храниться в регистре esp и по этому адресу располагается число равное количеству аргументов командной строки (включая имя программы) Посмотрите остальные позиции стека – по адресу [esp+4] располагается адрес в памяти где находится имя программы, по адресу [esp+8] храниться адрес первого аргумента, по адресу [esp+12] – второго и т.д (рис. 3.17).

```

palaciosfelipe@palaciosfelipe:~/work/arch-pc/lab09 — gdb --args lab09-3 аргумент1 аргумент2 аргумент3
Starting program: /home/palaciosfelipe/work/arch-pc/lab09/lab09-3 аргумент1 аргумент2 аргумент3
This GDB supports auto-downloading debuginfo from the following URLs:
  <https://debuginfod.fedoraproject.org/>
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
Downloading separate debug info for system-supplied DSO at 0xf7ffc000: skipped.
Breakpoint 1, 0x080490e8 in _start ()
(gdb) x/x $esp
0xffffd030: 0x00000005
(gdb) x/s *(void**)(esp + 4)
0xffffd1e9: "/home/palaciosfelipe/work/arch-pc/lab09/lab09-3"
(gdb) x/s *(void**)(esp + 8)
Undefined command: ".". Try "help".
(gdb) x/s *(void**)(esp + 8)
0xffffd219: "аргумент1"
(gdb) x/s *(void**)(esp + 12)
0xffffd22b: "аргумент2"
(gdb) x/s *(void**)(esp + 16)
0xffffd23c: "2"
(gdb) x/s *(void**)(esp + 20)

```

Рис. 3.17: позиции стека

6. Задание для самостоятельной работы

1.Преобразуйте программу из лабораторной работы No8 (Задание No1 для самостоятельной работы), реализовав вычисление значения функции $\text{fib}(x)$ как подпрограмму (рис. 3.18).

```

palaciosfelipe@palaciosfelipe lab09]$ touch lab9-F1.asm
palaciosfelipe@palaciosfelipe lab09]$ nasm -f elf lab9-F1.asm
palaciosfelipe@palaciosfelipe lab09]$ ld -m elf_i386 -o lab9-F1 lab9-F1.o
palaciosfelipe@palaciosfelipe lab09]$ ./lab9-F1
Результат: 25
palaciosfelipe@palaciosfelipe lab09]$

```

Рис. 3.18: 1

2. В листинге 9.3 приведена программа вычисления выражения $(3 + 2) \times 4 + 5$. При запуске данная программа дает неверный результат. Проверьте это. С помощью отладчика GDB, анализируя изменения значений регистров, определите ошибку и исправьте ее (рис. ??).

2

4 Выводы

Приобретены навыки написания программ с использованием подпрограмм. Ознакомлен с методами отладки при помощи GDB и его основными возможностями.

Список литературы